Laboratorul 1

Informatii generale de Windows API

Pentru a putea lucra cu windows api, trebuie sa includeti **<windows.h>**

Tipuri de date:

Pentru a pastra aceiasi dimensiune a datelor, indiferent de compilator sau platforma, in Windows API, sunt definite mai multe tipuri. O parte din cele mai des folosite, sunt urmatoarele:

- BYTE: echivalentul unui **unsigned char**, defineste un numar reprezentat pe un singur byte, fara semn
- WORD: echivalentul unui **unsigned short,** defineste un numar reprezentat pe 2 bytes, fara semn
- DWORD (double word): echivalentul unui **unsigned int,** defineste un numar reprezentat pe 4 bytes, fara semn
- STR: (nu prea e folosit): defineste un string de caracatere. (deja alocat)

Fiecare din tipurl definite mai sus, poate avea un prefix in fata: LP(long pointer) sau P (pointer), care defineste de fapt un pointer catre acel tip de date.
De ex: LPDWORD este echivalent cu DWORD*, mai exact, un pointer la un DWORD

Alt prefix folosit, este **C,** de la constant. Acest tip este folosit impreuna cu prefixul LP sau P pentru transmite compilatorului ca zona la care trimite pointerul, este una care nu se modifica (adica este constanta),
LPCSTR = const char *

Exista si tipul **LPVOID**, care este definit ca void*. Scopul acestui tip de date e pentru a transmite programatorului ca sistemul de operare asteapta un buffer (o zona de memorie) fara sa-l intereseze cum sunt date acolo. Datele vor fi transformate ulterior de programatorul in tipul necesar.

Unul din tipurile cel mai des intalnite, este **HANDLE:**

Handle-ul defineste deobicei accesul catre o resursa. Sistemul de operare tine un tabel cu fiecare resursa folosita, iar handle-ul este de fapt indexul in acea tabela. (tabela respectiva este in kernel, asa ca programatorul nu are acces la ea).

Obiectele de tip handle sunt returnate de functii care dau acces la resurse, iar lucrul cu obiecte de tip handle trebuie realizat doar prin intermediul altor functii de windows API. De exemplu, functia *fopen*, intoare un pointer catre un obiect de tip *FILE*, care apoi este folosit in alte functii, de exemplu, *fread*. Similar, functia CreateFile, intoarce un HANDLE, care apoi este folosit de alte functii, de exemplu, ReadFile.

Desi accesul la orice resursa din sistem este realizat prin intermediul unui obiect de tip HANDLE, fiecare HANDLE trebuie folosit doar in functii specifice obiectului. Mai exact, vom folosi functii pentru lucru cu fisiere atunci cand avem un HANDLE catre un fisier, si functii pentru procese, atunci cand avem un HANDLE catre un proces.

Pentru mai multe tipuri de date, consultati (https://msdn.microsoft.com/en-us/library/windows/desktop/aa383751(v=vs.85).aspx)

Folosirea functiilor pentru lucru cu fisiere

1. **Citirea, Scrierea, Stergerea…**

- **Deschiderea fisierului:**
  Ca si in stdio.h, pentru operatii cu fisiere, trebuie intai obtinut accesul la fisier. Daca in stdio.h, se obtinea un FILE*, in Windows API, se va obtine un HANDLE. Pentru aceasta, se va folosi functia CreateFile.

```
HANDLE WINAPI CreateFile(
  _In_      LPCTSTR LpFileName,
  _In_      DWORD dwDesiredAccess,
  _In_      DWORD dwShareMode,
  _In_opt_  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  _In_      DWORD dwCreationDisposition,
  _In_      DWORD dwFlagsAndAttributes,
  _In_opt_  HANDLE hTemplateFile
);
```

lpFileName	pointer catre numele fisierului
dwDesiredAccess	Tipul de acces catre fisier(citire, scriere)
dwShareMode	ce actiuni pot face alte procese, cat timp este deschis fisierul
lpSecurityAttributes	Drepturi de securitate specific, daca se doresc
dwCreationDisposition	Ce comportament sa aiba functia fisierul deja exista sau daca nu exista
dwFlagsAndAttributes	attribute pentru noul fisier
hTemplateFile	in caz ca se doreste utilziarea unui template pentru a seta atributele fisierului

Atentie! Chiar daca functia are *Create* in nume, nu este folosita doar pentru a crea un fisier, ci si pentru a-l deschide.

La fel ca toate functiile care returneaza un HANDLE catre un obiect, functia CreateFile va verifica intai daca utilizatorul din contextual carui ruleaza procesul care apeleaza aceasta functie, are accesul cerut. In caz ca nu poate deschide fisierul, functia va returna INVALID_HANDLE_VALUE.

- **Citirea/Scrierea fisierului**

  Handle-ul obtinut va fi utilizat pentru citire sau scriere, utilizand functiile *ReadFile* sau *WriteFile.* Cele doua functii au parametrii similari, asa ca voi descrie aici doar cei pentru ReadFile.

```
BOOL WINAPI ReadFile(
  _In_         HANDLE hFile,
  _Out_        LPVOID lpBuffer,
  _In_         DWORD nNumberOfBytesToRead,
  _Out_opt_    LPDWORD lpNumberOfBytesRead,
  _Inout_opt_  LPOVERLAPPED lpOverlapped
);
```

hFile	HANDLE-ul obtinut prin functia CreateFile
lpBuffer	Zona de memoria alocata in care sa puna datele citie din fisier
nNumberOfBytesToRead	Cati bytes sa incerce sa citeasca din fisier
lpNumberOfBytesRead	Cati bytes a reusit sa citeasca din fisier
lpOverlapped	O structura ce poate fi folosita pentru citire asincrona (pentru detalii, consultati documentatia sau intrebati la orele de laborator)

Functia ReadFile si WriteFile va returna TRUE, daca a putut citi din fisier (chiar daca a citit 0 bytes).

- **Inchiderea fisierului**

  Se realizeaza cu ajutorul functiei CloseHandle

```
BOOL WINAPI CloseHandle(
  _In_  HANDLE hObject
);
```

Un exemplu pentru citire si scriere, poate fi gasit la urmatorul link:
https://msdn.microsoft.com/en-us/library/windows/desktop/bb540534(v=vs.85).aspx

2. Enumerearea fisierelor si a directoarelor

Pentru a enumera fisierele dintr-un director, se folosesc urmatorii pasi:

- Se initializaza enumerarea, folosind functia *FindFirstFile:*

```
HANDLE WINAPI FindFirstFile(
  _In_   LPCTSTR lpFileName,
  _Out_  LPWIN32_FIND_DATA lpFindFileData
);
```

lpfileName	Path-ul in care se doreste sa se faca enumerarea. Path-ul poate contine * si ? pentru a filtra anumite fisiere(* inlocuieste un numar variabil de caractere,

	iar ? inlocuieste un singur caracter). De ex, daca doresc toate fisierele cu extensia txt din directorul C:\windows, lpFileName va fi: C:\windows*.txt, iar pentru toate fisierele din directorul C:\Windows, va fi : C:\windows*
lpFindFileData	Pointer catre o structura de tip WIN32_FIND_DATA (care a fost deja alocata). Aici se vor pune informatii despre primul fisier gasit care se potriveste sablonului dat la lpFileName

Daca functia reuseste, se va returna un HANDLE catre un obiect de enumerare de fisiere, ce poate fi folosit pentru a afla si informatii despre urmatorele fisiere.
In caz ca functia nu reuseste, va returna INVALID_HANDLE_VALUE

- Handle-ul primit de la FindFirstFile, va fi trimis, intr-o bucla (de obicei, do-while) la functia FindNextFile pentru a afla informatii despre urmatorul fisier.

```
BOOL WINAPI FindNextFile(
  _In_   HANDLE hFindFile,
  _Out_  LPWIN32_FIND_DATA lpFindFileData
);
```

De fiecare data cand se apeleaza functia FindNextFile, in lpFindFileData se vor pune informatii despre urmatorul fisier gasit care se potriveste sablonului de cautare dat la functia FindFirstFile. Cand cautarea se termina (nu mai exista fisiere conform regulii stabiliate la FindFirstFile), functia returneaza FALSE

Structura WIN32_FIND_DATA, arata in felul urmator

```
typedef struct _WIN32_FIND_DATA {
  DWORD    dwFileAttributes;
  FILETIME ftCreationTime;
  FILETIME ftLastAccessTime;
  FILETIME ftLastWriteTime;
  DWORD    nFileSizeHigh;
  DWORD    nFileSizeLow;
  DWORD    dwReserved0;
  DWORD    dwReserved1;
  TCHAR    cFileName[MAX_PATH];
  TCHAR    cAlternateFileName[14];
} WIN32_FIND_DATA, *PWIN32_FIND_DATA, *LPWIN32_FIND_DATA;
```

Dupa cum se observa, sunt destul de multe informatii despre fisier, cum ar fi (atributele, cand a fost creat/accesat/scris, dimensiunea si numele). Un Director, este tratat asemenea unui fisier. Pentru a vedea daca un fisier este director sau nu, se verifica atributele acestuia. Fiecare atribut, reprezinta un bit din cadrul unui numar pe 32 de biti (dwFileAttributes). Pentru a testa daca bitul pentru director este setata, trebuie verificat in cadrul acestui numar printr-o operatie binara.

```
if (lpFindFileData.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY !=0)
	//este director
Else
	//este fisier
```

Pentru a inchide obiect de enumerare de fisiere, se foloseste functia **FindClose.**

```
BOOL WINAPI FindClose(
  _Inout_  HANDLE hFindFile
);
```

Un exemplu de listare de fisiere, poate fi gasit la adresa: https://msdn.microsoft.com/en-us/library/windows/desktop/aa365200(v=vs.85).aspx

Alte functii utile ce lucreaza cu fisiere, sunt : DeleteFile, SetFilePointer, GetFileSize,CopyFile, MoveFile,SetFileAttributes, CreateDirectory, RemoveDirectory,GetCurrentDirectory, SetCurrentDirectory

Lista completa, poate fi gasita la adresa :
https://msdn.microsoft.com/en-us/library/windows/desktop/aa364232(v=vs.85).aspx
si
https://msdn.microsoft.com/en-us/library/windows/desktop/aa363950(v=vs.85).aspx